Power performance analysis of the iterative-MIMO adaptive switching algorithm detector on the FPGA hardware

Power Performance Analysis of the Iterative-MIMO Adaptive Switching Algorithm Detector on the FPGA Hardware

Nina Tadza and John S. Thompson and David I. Laurenson
Institute for Digital Communications
School of Engineering
University of Edinburgh, UK
Email: {n.tadza, john.thompson, dave.laurenson}@ed.ac.uk

***Abstract*—In this paper, a comprehensive power performance analysis of a novel Adaptive Switching Algorithm for an iterative-MIMO system is investigated with the prime goal of minimizing energy consumption in the receiver. The algorithm works by switching between a high performance detection method, the Fixed Sphere Decoding, and a much lower complexity algorithm, the Vertical-Bell Laboratories Layered Space-Time Zero Forcing technique, controlled by a threshold according to the mutual information calculated during each transmission. Results show significant improvements over current non-adaptive receivers, where energy savings of more than 60% can be obtained using on the latest Xilinx®Virtex-7 FPGA hardware.**

I. Introduction

Decoding received signals from an iterative-Multiple Input Multiple Output (MIMO) wireless system is computationally-expensive. Receiver performance is not only based on the success rate of the receiver recovering the data sent by the transmitter, but also in achieving this with minimal energy consumption. A system that could operate with low energy consumption whenever feasible is advantageous. Such detection algorithms currently active include Zero Forcing (ZF) with Decision Feedback (ZF-DF) [1], Sphere Decoder (SD) [2], Semidefinite-Relaxation (SDR) [3] etc. Although most work well in the detection process, they lack adaptivity, whereby, most detectors behave independently of the received signal characteristics and current channel conditions, which may waste computational resources. Several detection methods have been proposed to date to overcome this problem, however, one that could fit perfectly with MIMO characteristics has not been very well investigated. Most publications focus on saving power by using the Signal-to-Noise Ratio (SNR) [4], channel matrix condition number [5] or reducing the number of decoding iterations. These criteria are not enough to optimize the information on the entire MIMO setup. To tackle this, this paper considers the Mutual Information (MI) between the MIMO transmitters and receivers so that the diversity of the channels is fully exploited. Combining the MI with the noise level give better information regarding a channel in comparison to using either condition number or SNR alone.

This paper builds on the work of [7] by implementing the Adaptive Switching Algorithm onto an Field Programmable Gate Arrays (FPGA) hardware, in hope to gain further power and energy savings during hardware implementation. This gain is on top of the energy savings due to the switching receiver described in [7]. The FPGA is chosen as an exemplar platform for rapid prototyping purposes. Generally, even though it is more efficient to use an Application-Specific Integrated Circuit (ASIC) implementation, they usually require very long design times. FPGA is expected to produce generally similar trends and trade-offs with a fraction of the design time as other hardware platforms due to its re-programmability. This means it can provide a suitable platform for evaluating the implementation of the Adaptive Switching method in an iterative-MIMO system. The key to power savings comes from the algorithm exploiting the adaptivity in the detector according to the current conditions. The main contributions of this paper are summarized as follows:

- Realistic power and energy savings trends of the Adaptive Switching Algorithm are computed for example hardware circuitry.
- Detailed power analysis and the potential benefits of *Sleep Modes* and *Parallelization* as power savings techniques show more promising results in contrast to the *Voltage and Frequency Scaling*.

The rest of the paper is organized as follows; the Adaptive Switching Algorithm and its hardware design is given in Section II; Section III outlines several power saving methods evaluated in this paper; Section IV discusses the MIMO system under consideration; whilst the key findings are summarized in Section V; lastly, Section VI concludes the paper.

II. Adaptive Switching Algorithm

The Adaptive Switching Algorithm [7] is demonstrated with two well-known detection algorithms, namely the Fixed Sphere Decoder (FSD) [6], and the Vertical-Bell Laboratories Layered Space-Time [1] with the Zero Forcing (V-BLAST/ZF) technique. Switching between algorithms is determined by thresholds pre-calculated from the MI between the transmitter and the receiver, according to the real-time channel conditions.

A. Breakdown of Algorithm Design

1) V-BLAST/ZF: V-BLAST/ZF [1] is best deployed in high SNR environments, when the chances of successful decoding are high. Figure 1 illustrates the block diagram of the algorithm during implementation.

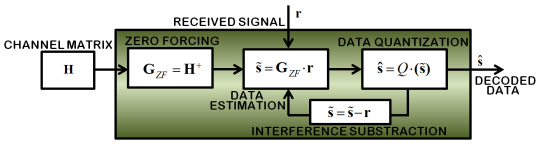

Fig. 1. Breakdown of V-BLAST/ZF Implementation Model

The algorithm minimizes the impact of noise by re-ordering the beamformer matrix, $\mathbf{G}$, which is the Moore-Penrose pseudoinverse of $\mathbf{H}$, with respect to the received signal strength. It processes the symbols, $\mathbf{r}$, according to this order i.e. handling the highest SNR antenna first. The signals are quantized to the nearest estimates, Q, using the quantizer function followed by linear combinatorial nulling and successive cancellation until all signals, $\hat{\mathbf{s}}$, are decoded.

2) FSD: The more complex detection method, the FSD, published in [6] can be viewed as running multiple V-BLAST/ZF detectors in parallel, each checking different transmit data combinations. Figure 2 provides the breakdown of the algorithm. The channel pseudoinverse, $\mathbf{G}$, is obtained by applying the $\mathbf{QR}$ decomposition to the channel matrix, $\mathbf{H}$.

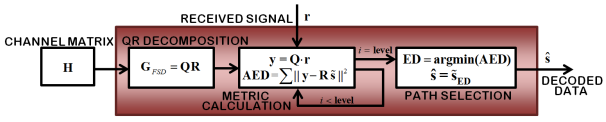

Fig. 2. Breakdown of FSD Implementation Model

The algorithm traverses down the tree, i, until the end of the tree i.e. the leaf is discovered, computing the Euclidean Distance (ED). FSD determines beforehand the number of nodes $\tilde{\mathbf{s}}$ around signal $\mathbf{r}$ that will be explored independent of the noise level, which means the search of an FSD is fixed for each candidate per antenna level. This yields an algorithm suitable for parallel implementation. The symbols $\hat{\mathbf{s}}$ associated with the minimum ED are the final solution.

3) Adaptive Switching Algorithm: The main idea behind the Adaptive Switching Algorithm is shown in Figure 3. The Threshold Control Block calculates the value of the accumulated MI, denoted by $\bar{I}$, obtained in the transmitter in relation to the receiver and activates the appropriate detector, either V-BLAST/ZF or FSD. Within the Threshold Control Block sits the MI calculation as shown in Equation (1).

$$\bar{I}(\mathbf{H}) \triangleq \log_2 \det \left(I + \frac{\mathbf{H}\mathbf{H}^T}{N_0} \right) \tag{1}$$

This calculation assumes the channel matrix $\mathbf{H}$ is perfectly known at the receiver with independent elements representing a block Rayleigh fading propagation environment, where T denotes the transpose operator and N_0 is the power of additive, independent and identically distributed (i.i.d.) circular symmetric complex Gaussian noise.

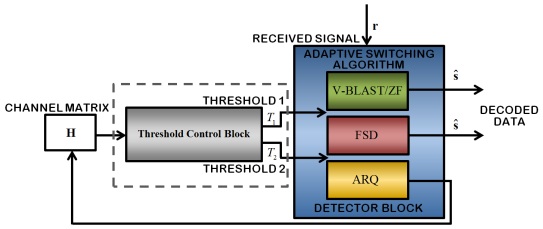

Fig. 3. Breakdown of Adaptive Switching Algorithm Implementation Model

The accumulated MI, $\bar{I}$, is dependent on the current channel conditions i.e. the noise level, N_0. The thresholds, T_1 and T_2, are pre-determined. If the MI computed is higher than the T_1 threshold, V-BLAST/ZF is chosen. FSD is selected when the transmitting environment is acceptable, which is when the MI value is in-between T_1 and T_2. When the channel is too poor for reliable recovery of the received signals, the detector block would send an Automatic Repeat reQuest (ARQ) for a re-transmission, avoiding forward error correction decoding when this is expected to fail, however, formally characterizing this decoding effect is out of scope of the present paper.

III. Power Saving Techniques

This paper investigates several power saving techniques when the Adaptive Switching Algorithm is implemented in FPGA hardware.

A. Voltage and Frequency Scaling

The power and energy consumption of a circuit depends on the number of computations performed over a fixed duration. By lowering the number of computations and varying the supply voltage to lower the internal clock frequency of the chip at run-time, the overall power consumption is lowered. The basic principle detailed in [8] states that the power consumed by running the operation at a slower speed is less than to run it at full power and finishing early. This study considers only the dynamic power however, discarding other components of power such as leakage, idle, overhead, static as well as the power needed to activate the chip. This present paper attempts to take all power components into consideration.

B. Sleep Mode

Sleep Mode is when the electronics operate in idle mode with a very low power consumption so they appear switched off for a certain period. When calculations do not possess the same task length and/or processing speed, they do not finish processing at the same time, meaning for some proportion of the time, processor cores need not be active at all times. Therefore, switching off the cores could be a means of saving power. By running the application as fast as possible, longer *Sleep Modes* can be deployed. This is a direct contradiction

to the findings of [8], where the reduction in dynamic power is inferior to the savings gained by scaling above. This paper attempts to discover, which power savings mode is best when other power components are also considered.

C. Parallelization

Part of optimizing a system in current chip designs is to construct the algorithms in such a way that parallel operations are possible. Using multiple processors provide a trade-off between utilizing more chip space and increasing the throughput of the algorithm. The cores split and share the computational load evenly amongst them. Therefore, each core performs only a fraction of the total computation depending on the number of cores available [8]. Furthermore, hardware architectures that can perform multiple tasks slowly in parallel should be more power efficient in comparison to computing a single operation at a higher clock speed [9]. Therefore, this paper will study how to combine the level of *Parallelization* with *Voltage and Frequency Scaling* technique.

IV. Experimental Setup

The experiment uses a software/hardware setup using Matlab™ and Xilinx® System Generator. The iterative-MIMO system under consideration comprises $M = 4$ transmitters and $N = 4$ receivers based on a Bit-Interleaved Coded Modulation (BICM) setup, which has a transmit frame size of K_u = 1,024 bits transmitting over a random independent Rayleigh fading propagation channel, $\mathbf{H}$, with independent fading elements, which is perfectly known at the receiver.

The transmitted bits, K_u, are encoded using an iterative-turbo scheme at rate of $R_c = 1/2$, which are then interleaved randomly to give, K_a coded bits, before mapping into a Quadrature Amplitude Modulation (QAM) constellation, $\mathcal{O}$, of $J = 16$, forming a sequence of $K_s = K_e/\log_2 J$ symbols. The 512 symbols are divided equally between the transmitters for 100,000 channel realizations. This part of the system is simulated using Matlab™. At the receiver, where the focus of this paper lies, is where the Adaptive Switching Algorithm detector is implemented using Xilinx®Virtex-7 chip. The receiver FPGA implementation is obtained using the Xilinx® System Generator.

V. Results and Findings

The total resource allocation of the Adaptive Switching Algorithm is given in Table I. The power usage can be calculated

TABLE I
Resource Allocation of the Adaptive Switching Algorithm

XILINX®VIRTEX-7 : XC7VLX330TFFG1157

Logic Resource Utilization	Used	Available	Utilization
Slice Registers	12,528	408,000	3%
Flip Flops	4,361	51,000	8%
4-Input LUTs	11,429	204,000	5%
DSP48E	132	1,120	11%
Memory (RAM)	41	1,500	2%

when the algorithm is implemented on the Xilinx® System Generator using the Xilinx® XPA™ tool. The power readings specified by the tool is generally dominated by the dynamic and static power terms, where dynamic is the power spent within a chip due to toggling of transistors, the value of voltage, the capacitance and is a function of the FPGA clock frequency. Static power is consumed due to transistor leakage and is highly dependent on the manufacturing process, the ambient temperature of the circuit, and the operating voltage. In order to determine the effectiveness of the algorithm, instead of power, a better parameter to consider is the energy, which is the power multiplied by the processing time. This information gives a better understanding of the system's efficiency in transferring the same size data packets within an allocated amount of time. Since this paper studies the energy efficiency of the system instead of maximizing the throughput, it is assumed that the system adopts low channel utilization policy, where packets are decoded at a maximum time of 20 μs.

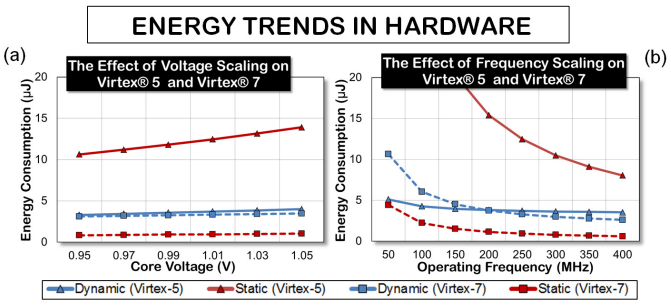

Fig. 4. Energy Trends with (a) the voltage applied and (b) the variation of clock frequencies on the Xilinx®Virtex-5 and Virtex-7 respectively

The energy trends are shown in Figure 4, where the dynamic and static energy consumption are compared on Xilinx®Virtex-5 [7] and Virtex-7. By comparing Figure 4(a), similar trends for scaling up the voltage in both chips can be observed, whereby, the energy is directly proportional to the voltage. When comparing the frequency however shown in 4(b), the energy consumption decreases with every frequency increment. From now on, the only chip under consideration is the Xilinx®Virtex-7. First, the main difference to note here is that dynamic energy dominates and therefore, the *Voltage and Frequency Scaling* may be able to save power in the detector [8]. Secondly, "high performance" and "low power" modes can be obtained by taking the extreme ends of the scaling ranges. If running the algorithm at the highest possible mode would save power, then the *Sleep Mode* would be an energy efficient method for the algorithm. Lastly, due to the small percentage of the area utilization, summarized in Table I, the algorithm has the potential for *Parallelization*, i.e. having multiple copies of the detector. This paper attempts to instigate the three techniques mentioned in Section III and determines if they might increase energy savings.

A. Voltage and Frequency Scaling

Figure 4 shows that, due to the higher level of dynamic to static energy, where it is approximately six times larger, the

overall energy of the circuit can be optimized. However, when considering the total energy of the chip, this might no longer be the case.

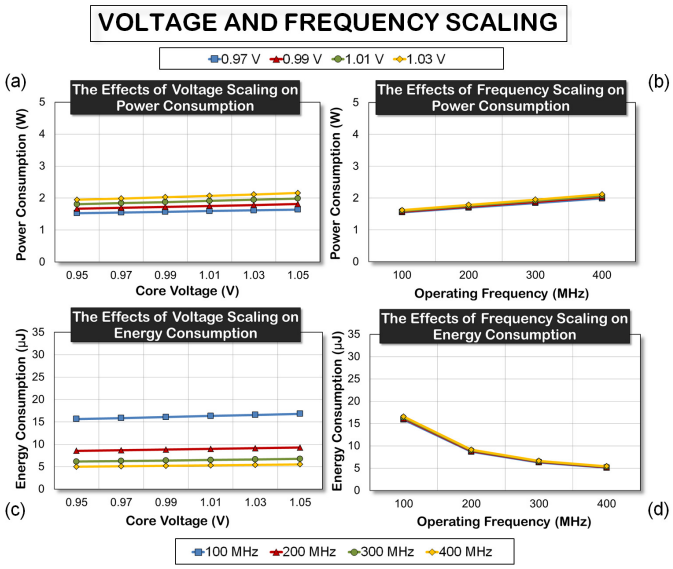

Fig. 5. Voltage and Frequency Scaling Effects where (a) and (b) are with voltage applied, (c) and (d) are with the variation of frequencies respectively

Figure 4(b) confirms this as the energy required to run the task at 400 MHz is less than 0.6 μJ in comparison to 2.9 μJ at 100 MHz, giving a difference of more than 65%. From this, it can be said that running the algorithm as quickly as possible at the lowest possible voltage and switching it off would be better than running it at a slower clock speed. The total power and energy consumption during the *Voltage and Frequency Scaling* are given in Figure 5. Similar to the previous experiment, the scaling of voltage is proportional to the power and energy consumption, which can be seen in Figure 5(a) and 5(c). Taking a clock speed of 200 MHz as an example, at voltages of 0.97 V and 1.03 V, the latter gives an increased power usage of 12%. Though minimal, it is still an undesired result. In contrast, Figure 5(b) illustrates that even with a minimal increment of power in frequency scaling, the reduction in energy shown in Figure 5(d) is substantial. Looking at a voltage of 0.99 V, running the algorithm four times faster provides 51% energy savings.

Moreover, Figure 5(d) shows the total energy required to decode the same packet of data is less, due to the faster decoding process. It suggests that running the algorithm at full speed would be better than to finish processing just in time. This means that instead of having it running at lower power and taking the maximum 20 μs to decode the data packet, the system would finish processing in less than 3 μs and be put into *Sleep Mode* for 78% of the time. This concludes that voltage scaling is not suitable as a power savings technique for the Adaptive Switching Algorithm on an architecture where static power is a significant component of power consumption.

B. Sleep Mode

Taking the extreme cases of the chip's lower and upper limit of voltage and frequency operations into consideration, "low power" and "high performance" modes can be evaluated. Table II reviews the parameters of the Xilinx®Virtex-7 when running the Adaptive Switching Algorithm in two separate modes. The power usage analyzed by the Xilinx® XPA™ tool are given as 1.5 W and 2.2 W for low power and high performance modes respectively, contributing to 19% increase in power usage when high performance mode is selected. The total maximum energy saving is equivalent to 69%.

TABLE II
LOW POWER AND HIGH PERFORMANCE PARAMETERS

Operation Mode/ Parameters	Low Power	High Performance
Core Voltage	0.97 V	1.03 V
Operating Frequency	60 MHz	400 MHz
Max Throughput	240 Mbps	1200 Mbps
Total Power Consumption	-	19%
Total Energy Savings	-	69%

This section confirms the previous conclusion where, it takes less energy to transfer the same data packet in "high performance" mode. Therefore, by running the algorithm as fast as possible and then switching the cores off would save more energy, and thus, *Sleep Modes* are a good way to save energy (and power) in the detector.

C. Parallelization

The Adaptive Switching Algorithm has quite low complexity and only uses a small percentage of the Xilinx®Virtex-7 as evident in Table I. This suggests promising results for parallel implementation, which are shown in Figure 6.

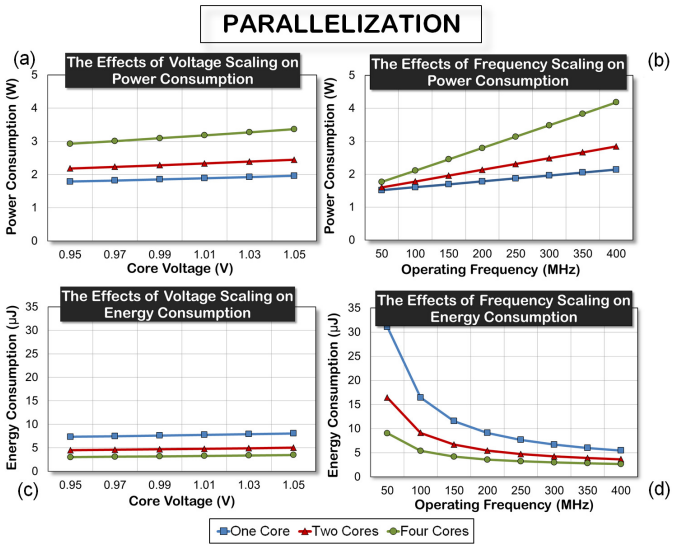

Fig. 6. Results for Parallel Implementation, (a) and (c) with the voltage applied, (b) and (d) with the variation of frequencies respectively

Multiple copies of the Adaptive Switching Algorithm are utilized with one core matching one copy of the algorithm

being used on the FPGA. As predicted, the more cores are used on the FPGA, the more power the chip needs as evident in Figure 6(a). This is due to the extra power needed to activate the multiple cores on the chip. However, the increase in power consumption is small, at maximum, 20%, with every quadruple number of cores used, which is evident at every voltage point. When it comes to energy however, although voltage scaling has little effect, the parallel setup does save significant overall energy savings seen in Figure 6(d).

TABLE III
"LOW POWER" AND "HIGH PERFORMANCE" PARALLEL IMPLEMENTATIONS

Number of Cores / Parameters	One		Two		Four	
	Low	High	Low	High	Low	High
Power Consumption	1.5 W	2.2 W	1.6 W	2.9 W	1.7 W	4.3 W
Energy Consumption	30.4 µJ	5.6 µJ	16.0 µJ	3.8 µJ	8.8 µJ	2.8 µJ
Power Usage	-	19%	3%	32%	6%	48%
Energy Savings	-	69%	31%	78%	55%	83%

Low Power: *0.97V, 60MHz*; **High Performance:** *1.03V, 400MHz*

The same can be said in frequency scaling, evident in Figure 6(b) and Figure 6(d), for power and energy respectively, where, taking frequency of 200 MHz as an example, running four cores instead of one gives 42% energy savings with only a 14% increase in power. The energy saved whilst running parallel cores in comparison to running a single thread is substantial, ranging from 3% to 68% across all frequencies, with particularly large differences at lower clock frequencies. These results show that, *Parallelization* is a good way to minimize the energy consumption.

A combination of the techniques is evaluated to see if more energy savings can be gained. Table III summarizes the parameters of the power consumption and energy savings when the algorithm is run in parallel on "low power" and "high performance" modes, calculated against the "low power", single core baseline. The "low power" mode in fact uses more energy to process the same data packet in comparison to the "high performance" mode. Moreover, *Parallelization* offers significant energy savings regardless of which mode is on, with a minimal increase in power to activate the extra cores. For example, by using four cores, in "low power" mode, the single core design uses 55% more energy than its multicore counterpart, with only a 6% increase in power.

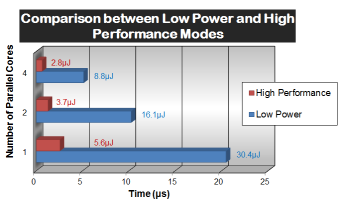

Fig. 7. Modes Comparison on Parallel Implementation

Figure 7 shows the energy used and time needed to decode the data packet received. These can be calculated from the power usage listed in Table III. *Parallelization* causes the chip to use less energy on four cores, giving a total energy savings of 55% and 33% for considering separately the "low power" and "high performance" modes respectively. With these results, it can be concluded that, the more cores deployed, the more energy efficient the Adaptive Switching Algorithm becomes. Instead of having one core running the algorithm for the entire 20 μs, using four cores running at for a quarter of the duration, and shutting them off for 75% of the time would minimize the energy consumption. Furthermore, the more cores being utilized, the more energy can be saved. When combining *Voltage and Frequency Scaling* and *Parallelization* techniques, i.e. comparing one core "low power" mode and "high performance" multicore mode, with energy values of 30.4 μJ and 2.8 μJ respectively, saves a total of more than 80%. This shows that combining the two saving techniques achieves significant combined energy savings.

VI. CONCLUSION

In contrast that running the detector at a slower speed would improve energy consumption [8], when considering the overall power usage, i.e. dynamic and static, the results obtained for the Xilinx®Virtex-7 recommend the Adaptive Switching Algorithm to be run as fast as possible and be put into *Sleep Mode*. Additionally, the benefits of voltage scaling are not significant as the limited voltage scaling range gives a negligible difference in energy consumption. On the other hand, the frequency scaling suggests that the algorithm works best when running at the highest frequency so that it can be put into *Sleep Mode* sooner, conserving energy. In addition, the more cores are used, the faster the task completion, the faster it can be put into idle mode, thus saves significant energy, where more than 60% can be saved.

ACKNOWLEDGMENT

NT is funded by the Universiti Tun Hussein Onn Malaysia.

REFERENCES

[1] G. D. Golden et al., *"Detection Algorithm and Initial Laboratory Results Using V-BLAST Space Time Communication Architecture"*, IEEE Elec. Letters, Vol. 35, No. 1, pp. 14, 1999.
[2] E. Viterbo et al., *"A Universal Lattice Code Decoder for Fading Channels,"*, IEEE Trans. Info Thy, Vol. 45, pp. 1639, 1999.
[3] P. Tan et al., *"The Aplication of Semidefinite Programming for Detection in CDMA,"*, IEEE JSAC, Vol. 19, pp. 1442, 2001.
[4] X. Wu et al., *"A Fixed-Complexity Soft-MIMO Detector via Parallel Candidate Adding Scheme and its FPGA Implementation"*, IEEE Comms. Letters, Vol. 15, No. 2, pp. 241, 2011.
[5] M. Matthaiou et al., *"Reduced Complexity Detection for Ricean MIMO Channels Based on Channel Number Thresholding"*, IEEE PIMRC, pp. 1718, 2008.
[6] L. G. Barbero et al., *"Fixing the Complexity of the Sphere Decoder for MIMO Detection"*, IEEE Trans. Wireless Comms., Vol. 7, No. 6, pp. 2131, 2008.
[7] N. Tadza et al., *"Adaptive Switching Detection Algorithm for Iterative-MIMO Systems to Enable Power Savings"*, accepted by the AGU Radio Science Journal in Aug. 2014. A copy of the journal can be accessed at this website, http://edin.ac/1pomwMd
[8] E. G. Larsson et al., *"The Impact of Dynamic Voltage and Frequency Scaling on Multicore DSP Algorithm Design"*, IEEE Signal Processing Mag., Vol. 28, No. 3, 2011.
[9] E. Seo et al., *"Energy Efficient Scheduling of Real-Time Tasks on Multicore Processors"*, IEEE Trans. Parallel Distributed Systems, Vol. 19, No. 11, pp. 1540, 2008.